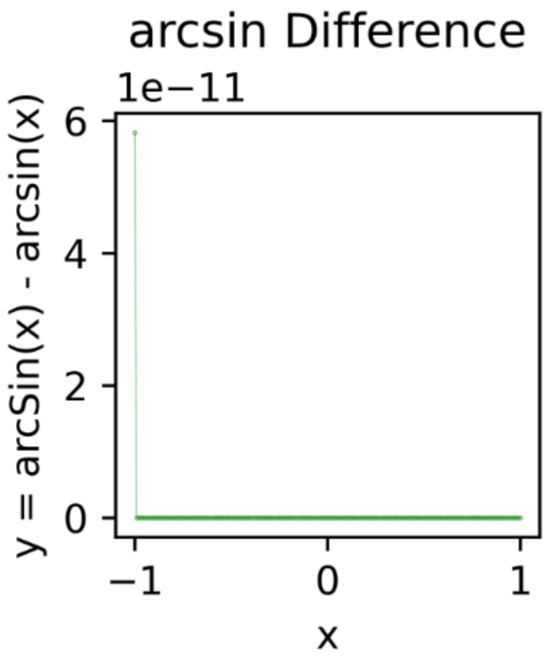
Nicholas Chan
nipchan@ucsc.edu
4/18/2021
CSE13s Spring ‘21

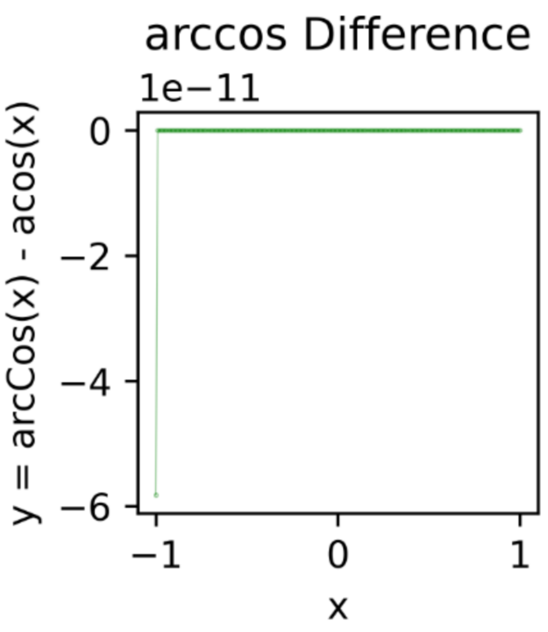
Writeup:
Assignment 2: Math Library

For assignment 2, I was comparing the performance of my implementation of the arcsin, arccos, arctan and log with the performance of the math.h libraries’ implementations of those same functions.

Below are the graphs of differences with the x axis representing the domain of inputs applied to these functions and the y axis representing the differences calculated by subtracting the outputs of my functions with the outputs of the math.h library’s functions.



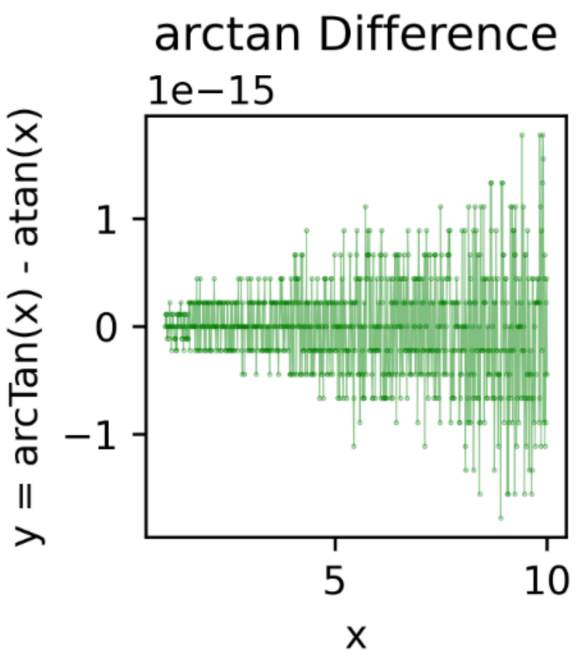
Graph of the difference in performances between my implementations and math.h’s implementations of arcsin and arccos



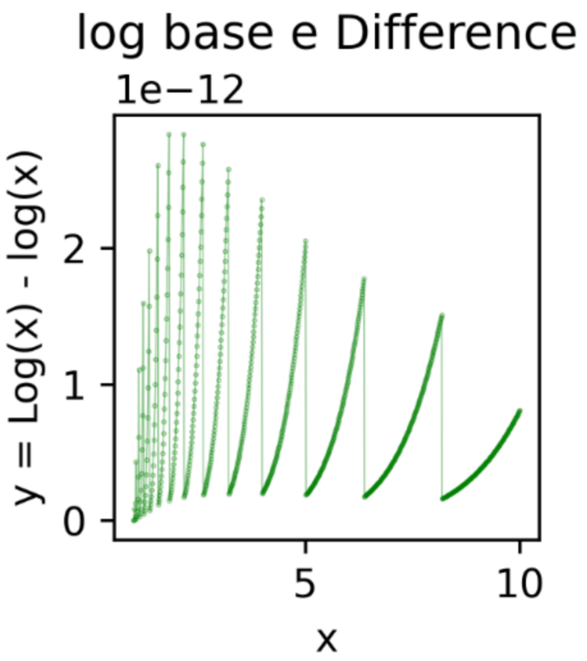
Graph of the difference in performances between my implementations and math.h's implementations of arccos

The first functions I implemented were arcsin and arccos. The magnitudes of the differences between my functions and the math.h functions were desirable up until x values near -1 and 1. I believe the cause for this sharp spike in error is due to my algorithm's inability to properly handle these boundaries which are known to yield asymptotically increasing values for arcsin. Although I used Newton's method to find the roots of translated sines and cosines, I am still a little confused about how exactly these boundary conditions of -1 and 1 confounded my function outputs so greatly. These two functions are explicitly linked because I used the identity connecting arcsin to arccos which was provided in the instructions for assignment 2. Because of these large spikes in difference, the magnitude that the graphs of arcsin and arccos get scaled up to is now 1e-11 when the vast majority of their differences are in fact around a magnitude of 1e-16. It is invisible from this graph but for all differences between -1 and 1 non inclusively, the differences, although small at a magnitude of around 1e-16, oscillate. This pattern of oscillation is likely due to my usage of Newton's Method which I will talk more about as it is seen even more clearly in my implementations for arctan and log. I believe the largest error I get is 5e-11, but it gets rounded up to 1e-10 in my test harness. This is for both arcsin and arccos since they are connected. I tried implementing the trig identity to overcome the hurdle of the asymptotic behavior of arcsin and arccos around -1 and 1.

The next functions I implemented were arctan and log base e. The magnitudes of the difference between my function and the math.h function were desirable throughout the test interval of 1 to 10. As shown in the graphs below, the magnitude of differences in arctan were at most 1e-15 while the magnitudes of log base e were at most 1e-12. The trend in recorded differences of log base e's graph looked nice as it seemed like the difference between implementations was moving towards 0 as x increased. However arctan’s trend appeared a little concerning as it seemed to oscillate with a greater and greater amplitude as x increased. I believe that the unusual appearances of these graphs is due to the ways I used Newton's method. Newton’s method is the only thing that I can link between my arctan and log functions and my arcsin and arccos functions (arcsin and arccos are explicitly connected).



Graphs of the difference in performances between my implementations and math.h’s implementations of arctan



Graphs of the difference in performances between my implementations and math.h's implementations of log

The oscillations observed in the differences between my implementations and math.h's implementations of arctan and log would make sense as my implementation of Newton's Method would force my function to continue until the absolute value of my iterate became less than a specified value for epsilon. This condition would offer the iterates used in my functions for approximating arctan and log a lot of freedom to switch signs and magnitudes as only the absolute value of them will trigger a stop for iteration. This uncertainty in this algorithm would ultimately result in my approximation frequently switching off between underestimation and overestimation of the actual function I am trying to approximate.

Because of the variation in the degrees of success seen in my functions I am sure that arcsin and arccos have an additional hurdle to pass for approximating correctly with Newton's Method. The results of my implementations for arctan and log are magnitudes better than they are for arcsin and arccos likely due to the additional hurdle of asymptotic behavior around 1 and -1 for arcsin and arccos.

I wrote a matplotlib script to graph these results. Points for these graphs of difference over x value were taken at increments of 0.01.